

# Fast Ridge Regression with Randomized Principal Component Analysis and Gradient Descent

Yichao Lu <sup>\*</sup> and Dean P. Foster <sup>†</sup>

Department of Statistics  
Wharton, University of Pennsylvania  
Philadelphia, PA, 19104-6340

## Abstract

We propose a new two stage algorithm LING for large scale regression problems. LING has the same risk as the well known Ridge Regression under the fixed design setting and can be computed much faster. Our experiments have shown that LING performs well in terms of both prediction accuracy and computational efficiency compared with other large scale regression algorithms like Gradient Descent, Stochastic Gradient Descent and Principal Component Regression on both simulated and real datasets.

## 1 Introduction

Ridge Regression (RR) is one of the most widely applied penalized regression algorithms in machine learning problems. Suppose  $\mathbf{X}$  is the  $n \times p$  design matrix and  $\mathbf{Y}$  is the  $n \times 1$  response vector, ridge regression tries to solve the problem

$$\hat{\beta} = \arg \min \|\mathbf{X}\hat{\beta} - \mathbf{Y}\|^2 + n\lambda\|\hat{\beta}\|^2 \quad (1)$$

which has an explicit solution

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + n\lambda)^{-1} \mathbf{X}^\top \mathbf{Y} \quad (2)$$

However, for modern problems with huge design matrix  $\mathbf{X}$ , computing (2) costs  $O(np^2)$  FLOPS. When  $p > n \gg 1$  one can consider the dual formulation of (1) which also has an explicit solution as mentioned in (Lu et al., 2013; Saunders et al., 1998) and the cost is  $O(n^2p)$  FLOPS. In summary, trying to solve (1) exactly costs  $O(np \min\{n, p\})$  FLOPS which can be very slow.

There are faster ways to approximate (2) when computational cost is the concern. One can view RR as an optimization problem and use Gradient Descent (GD) which

takes  $O(np)$  FLOPS in every iteration. However, the convergence speed for GD depends on the spectral of  $\mathbf{X}$  and  $\lambda$ . When  $\mathbf{X}$  is ill conditioned, GD requires a huge number of iterations to converge which makes it very slow. For huge datasets, one can also apply stochastic gradient descent (SGD) (Zhang, 2004; Johnson and Zhang, 2013; Bottou, 2010), a powerful tool for solving large scale optimization problems.

Another alternative for regression on huge datasets is Principle Component Regression (PCR) as mentioned in (Artemiou and Li, 2009; Jolliffe, 2005), which runs regression only on the top  $k_1$  principal components of the  $\mathbf{X}$  matrix. PCA for huge  $\mathbf{X}$  can be computed efficiently by randomized algorithms like (Halko et al., 2011a,b). The cost for computing top  $k_1$  PCs of  $\mathbf{X}$  is  $O(npk_1)$  FLOPS. The connection between RR and PCR is well studied by (Dhillon et al., 2013). The problem of PCR is that when a large proportion of signal sits on the bottom PCs, it has to regress on a lot of PCs which makes it both slow and inaccurate.

In this paper, we propose a two stage algorithm LING<sup>1</sup> which is a faster way of computing the RR solution (2). A detailed description of the algorithm is given in section 2. In section 3, we prove that LING has the same risk as RR under the fixed design setting. In section 4, we compare the performance of PCR, GD, SGD and LING in terms of prediction accuracy and computational efficiency on both simulated and real data sets.

## 2 The Algorithm

### 2.1 Description of the Algorithm

LING is a two stage algorithm. The intuition of LING is quite straight forward. Note that regressing  $\mathbf{Y}$  on  $\mathbf{X}$  is essentially projecting  $\mathbf{Y}$  onto the span of  $\mathbf{X}$ . Let  $\mathbf{U}_1$  denotes the top  $k_2$  PCs (singular vectors) of  $\mathbf{X}$  and let  $\mathbf{U}_2$  denote the remaining PCs. The projection of  $\mathbf{Y}$  onto the span of  $\mathbf{X}$  can be decomposed into two orthogonal parts, the pro-

<sup>\*</sup> yichaolu@wharton.upenn.edu

<sup>†</sup> foster@wharton.upenn.edu

<sup>1</sup>LING is the Chinese of ridge

---

**Algorithm 1** LING

---

**Input :** Data matrix  $\mathbf{X}, \mathbf{Y}$ .  $\mathbf{U}_1$ , an orthonormal matrix consists of top  $k_2$  PCs of  $\mathbf{X}$ .  $d_1, d_2, \dots, d_{k_2}$ , top  $k_2$  singular values of  $\mathbf{X}$ . Regularization parameter  $\lambda$ , an initial vector  $\hat{\gamma}_{2,0}$  and number of iterations  $n_2$  for GD .

**Output :**  $\hat{\gamma}_{1,s}, \hat{\gamma}_2$ , the regression coefficients.

1. Regress  $\mathbf{Y}$  on  $\mathbf{U}_1$ , let  $\hat{\gamma}_1 = \mathbf{U}_1^\top \mathbf{Y}$ .
  2. Compute the residual of previous regression problem. Let  $\mathbf{Y}_r = \mathbf{Y} - \mathbf{U}_1 \hat{\gamma}_1$ .
  3. Compute the residual of  $\mathbf{X}$  regressing on  $\mathbf{U}_1$ . Use  $\mathbf{X}_r = \mathbf{X} - \mathbf{U}_1 \mathbf{U}_1^\top \mathbf{X}$  to denote the residual of  $\mathbf{X}$ .
  4. Use gradient descent with optimal step size with initial value  $\hat{\gamma}_{2,0}$  (see algorithm 3) to solve the RR problem  $\min_{\hat{\gamma}_2 \in \mathcal{R}^p} \|\mathbf{X}_r \hat{\gamma}_2 - \mathbf{Y}_r\|^2 + n\lambda \|\hat{\gamma}_2\|^2$ .
  5. Compute a shrinkage version of  $\hat{\gamma}_1$  by  $(\hat{\gamma}_{1,s})_i = \frac{d_i^2}{d_i^2 + n\lambda} (\hat{\gamma}_1)_i$
  6. The final estimator is  $\hat{\mathbf{Y}} = \mathbf{U}_1 \hat{\gamma}_{1,s} + \mathbf{X}_r \hat{\gamma}_2$ .
- 

jection onto  $\mathbf{U}_1$  and the projection onto  $\mathbf{U}_2$ . In the first stage, we pick a  $k_2 \ll p$  and the projection onto  $\mathbf{U}_1$  can be computed directly by  $\hat{\mathbf{Y}}_1 = \mathbf{U}_1 \mathbf{U}_1^\top \mathbf{Y}$  which is exactly the same as running a PCR on top  $k_2$  PCs. For huge  $\mathbf{X}$ , computing the top  $k_2$  PCs exactly is very slow, so we use a faster randomized SVD algorithm for computing  $\mathbf{U}_1$  which is proposed by (Halko et al., 2011a) and described below. In the second stage, we first compute  $\mathbf{Y}_r = \mathbf{Y} - \hat{\mathbf{Y}}_1$  and  $\mathbf{X}_r = \mathbf{X} - \mathbf{U}_1 \mathbf{U}_1^\top \mathbf{X}$  which are the residual of  $\mathbf{Y}$  and  $\mathbf{X}$  after projecting on  $\mathbf{U}_1$ . Then we compute the projection of  $\mathbf{Y}$  onto the span of  $\mathbf{U}_2$  by solving the optimization problem  $\min_{\hat{\gamma}_2 \in \mathcal{R}^p} \|\mathbf{X}_r \hat{\gamma}_2 - \mathbf{Y}_r\|^2$  with GD (Algorithm 3). Finally, since RR shrinks the projection of  $\mathbf{Y}$  onto  $\mathbf{X}$  via regularization, we also shrink the projections in both stages accordingly. Shrinkage in the first stage is performed directly on the estimated regression coefficients and shrinkage in the second stage is performed by adding a regularization term to the optimization problem mentioned above. Detailed description of LING is shown in Algorithm 1.

**Remark 1.** LING can be regarded as a combination of PCR and GD. The first stage of LING is a very crude estimation of  $\mathbf{Y}$  and the second stage adds a correction to the first stage estimator. Since we don't need a very accurate estimator in the first stage it suffices to pick a very small  $k_2$  in contrast with the  $k_1$  PCs needed for PCR. In the second stage, the design matrix  $\mathbf{X}_r$  is a much better conditioned matrix than the original  $\mathbf{X}$  since the directions with largest singular value have been removed. As introduced in section 2.2, Algorithm 3 converges much faster with a better conditioned matrix. Hence GD in the second stage of LING converges faster than directly applying GD for solving equation 1. The above property guarantees that LING is both fast and accurate compared with PCR and GD. More details about on the computational cost will be discussed in section 2.2.

---

**Algorithm 2** Random SVD

---

**Input :** design matrix  $\mathbf{X}$ , target dimension  $k_2$ , number of power iterations  $i$ .

**Output :**  $\mathbf{U}_1 \in n \times k_2$ , the matrix of top  $k_2$  left singular vectors of  $\mathbf{X}$ ,  $d_1, d_2, \dots, d_{k_2}$ , the top  $k_2$  singular values of  $\mathbf{X}$ .

1. Generate random matrix  $R_1 \in p \times k_2$  with i.i.d standard Gaussian entries.
  2. Estimate the span of top  $k_2$  left singular vectors of  $\mathbf{X}$  by  $A_1 = (\mathbf{X}\mathbf{X}^\top)^i \mathbf{X} R_1$ .
  3. Use QR decomposition to compute  $Q_1$  which is an orthonormal basis of the column space of  $A_1$ .
  4. Compute SVD of the reduced matrix  $Q_1^\top \mathbf{X} = \mathbf{U}_0 \mathbf{D}_0 \mathbf{V}_0^\top$ .
  5.  $\mathbf{U}_1 = Q_1 \mathbf{U}_0$  gives the top  $k_2$  singular vectors of  $\mathbf{X}$  and the diagonal elements of  $\mathbf{D}_0$  gives the singular values.
- 

---

**Algorithm 3** Gradient Descent with Optimal Step Size (GD)

---

**Goal :** Solve the ridge problem  $\min_{\hat{\gamma} \in \mathcal{R}^p} \|\mathbf{X}\hat{\gamma} - \mathbf{Y}\|^2 + n\lambda \|\hat{\gamma}\|^2$ .

**Input :** Data matrix  $\mathbf{X}, \mathbf{Y}$ , regularization parameter  $\lambda$ , number of iterations  $n_2$ , an initial vector  $\hat{\gamma}_0$

**Output :**  $\hat{\gamma}$

**for**  $t = 0$  **to**  $n_2 - 1$  **do**

$$Q = 2\mathbf{X}^\top \mathbf{X} + 2n\lambda \mathbf{I}$$

$$w_t = 2\mathbf{X}^\top \mathbf{Y} - Q\hat{\gamma}_t$$

$$s_t = \frac{w_t^\top w_t}{w_t^\top Q w_t}. \quad s_t \text{ is the step size which makes the target function decrease the most in direction } w_t.$$

$$\hat{\gamma}_{t+1} = \hat{\gamma}_t + s_t \cdot w_t.$$

**end for**

---

**Remark 2.** Algorithm 2 is essentially approximating the subspace of top left singular vectors by random projection. It provides a fast approximation of the top singular values and vectors for large  $\mathbf{X}$  when computing the exact SVD is very slow. Theoretical guarantees and more detailed explanations can be found in (Halko et al., 2011a). Empirically we find in the experiments, Algorithm 2 may occasionally generate a bad subspace estimator due to randomness which makes PCR perform badly. On the other hand, LING is much more robust since in the second stage it compensate for the signal missing in the first stage. In all the experiments, we set  $i = 1$ .

The shrinkage step (step 5) in Algorithm 1 is only necessary for theoretical purposes since the goal is to approximate Ridge Regression which shrinks the Least Square estimator over all directions. In practice shrinkage over the top  $k_2$  PCs is not necessary. Usually the number of PCs selected ( $k_2$ ) is very small. From the bias variance trade off perspective, the variance reduction gained from this shrinkage step is at most  $O(\frac{k_2}{n})$  under the fixed design setting (Dhillon et al., 2013) which is a tiny number.

Moreover, since the top singular values of  $\mathbf{X}^\top \mathbf{X}$  are usually very large compared with  $n\lambda$  (since large  $\lambda$  will introduce large bias), the shrinkage factor  $\frac{d_i^2}{d_i^2 + n\lambda}$  will be pretty close to 1 for top singular values. We use shrinkage in Algorithm 1 because the risk of the shrinkage version of LING is exactly the same as RR as proved in section 3.

Algorithm 2 can be further simplified if we skip the shrinkage step mentioned in previous paragraph. Instead of computing the top  $k_2$  PCs, the only thing we need to know is the subspace spanned by these PCs since the first stage is essentially projecting  $\mathbf{Y}$  onto this subspace. In other words, we can replace  $\mathbf{U}_1$  in step 1, 2, 3 of Algorithm 1 with  $\mathbf{Q}_1$  obtained in step 3 of Algorithm 2 and directly let  $\hat{\mathbf{Y}} = \mathbf{Q}_1 \hat{\gamma}_1 + \mathbf{X}_r \hat{\gamma}_2$ . In the experiments of section 4 we use this simplified version.

## 2.2 Computational Cost

We claim that the cost of LING is  $O(np(k_2 + n_2))$  where  $k_2$  is the number of PCs used in the first stage and  $n_2$  is the number of iterations of GD in the second stage. According to (Halko et al., 2011a), the dominating step in Algorithm 2 is computing  $(\mathbf{X}\mathbf{X}^\top)^i \mathbf{X} \mathbf{R}_1$  and  $\mathbf{Q}_1^\top \mathbf{X}$  which costs  $O(npk_2)$  FLOPS. Computing  $\hat{\gamma}_1$  and  $\mathbf{Y}_r$  cost less than  $O(npk_2)$ . Computing  $\mathbf{X}_r$  costs  $O(npk_2)$ . So the computational cost before the GD step is  $O(npk_2)$ . For the GD stage, note that in every iteration  $\mathbf{Q}$  never need to be constructed explicitly. While computing  $w_t$  and  $s_t$ , always multiply matrix and vector first gives a cost of  $O(np)$  for every iteration. So the cost for GD stage is  $O(n_2 np)$ . Add all pieces together the cost of LING is  $O(np(k_2 + n_2))$  FLOPS.

Let  $n_1$  be the number of iterations required for solving (1) directly by GD and  $k_1$  be the number of PCs used for PCR. Easy to check that the cost for GD is  $O(n_1 np)$  FLOPS and the cost for PCR is  $O(npk_1)$ . As mentioned in remark 1, the advantage of LING over GD and PCR is that  $k_1$  and  $n_1$  might have to be really large to achieve high accuracy but much smaller values of the pair  $(k_2, n_2)$  will work for LING.

Consider the case when the signal is widely spread among all PCs (the projection of  $\mathbf{Y}$  onto the bottom PCs of  $\mathbf{X}$  is large) instead of concentrating on the top ones,  $k_1$  needs to be large to make PCR perform well since all the signals on bottom PCs are discarded by PCR. But LING doesn't need to include all the signals in the first stage regression since the signal left over will be estimated in the second GD stage. Therefore LING is able to recover most of the signal even with a small  $k_2$ .

In order to understand the connection between accuracy and number of iterations in Algorithm 3, we state the following theorem in A.Epelman (2007):

**Theorem 1.** Let  $g(z) = \frac{1}{2}z^\top Mz + q^\top z$  be a quadratic function where  $M$  is a PSD matrix. Suppose  $g(z)$  achieves

minimum at  $z^*$ . Apply Algorithm 3 to solve the minimization problem. Let  $z_t$  be the  $z$  value after  $t$  iterations, then the gap between  $g(z_t)$  and  $g(z^*)$ , the minimum of the objective function satisfies

$$\frac{g(z_{t+1}) - g(z^*)}{g(z_t) - g(z^*)} \leq C = \left( \frac{A - a}{A + a} \right)^2 \quad (3)$$

where  $A, a$  are the largest and smallest eigenvalue of the  $M$  matrix.

Theorem 1 shows that the sub optimality of the target function decays exponentially as the number of iterations increases and the speed of decay depends on the largest and smallest singular value of the PSD matrix that defines the quadratic objective function. If we directly apply GD to solve (1), Let  $f_1(\beta) = \|\mathbf{X}\beta - \mathbf{Y}\|^2 + n\lambda\|\beta\|^2$ . Assume  $f_1$  reaches its minimal at  $\hat{\beta}$ . Let  $\hat{\beta}_t$  be the coefficient after  $t$  iterations and let  $d_i$  denote the  $i^{th}$  singular value of  $\mathbf{X}$ . Apply theorem 1 we have

$$\frac{f_1(\hat{\beta}_{t+1}) - f_1(\hat{\beta})}{f_1(\hat{\beta}_t) - f_1(\hat{\beta})} \leq C = \left( \frac{d_1^2 - d_p^2}{d_1^2 + d_p^2 + 2n\lambda} \right)^2 \quad (4)$$

Similarly for the second stage of LING, Let  $f_2(\gamma_2) = \|\mathbf{X}_r \gamma_2 - \mathbf{Y}_r\|^2 + n\lambda\|\gamma_2\|^2$ . Assume  $f_2$  reaches its minimal at  $\hat{\gamma}_2$ . We have

$$\frac{f_2(\hat{\gamma}_{2,t+1}) - f_2(\hat{\gamma}_2)}{f_2(\hat{\gamma}_{2,t}) - f_2(\hat{\gamma}_2)} \leq C = \left( \frac{d_{k_2+1}^2}{d_{k_2+1}^2 + 2n\lambda} \right)^2 \quad (5)$$

In most real problems, the top few singular values of  $\mathbf{X}^\top \mathbf{X}$  are much larger than the other singular values and  $n\lambda$ . Therefore the constant  $C$  obtained in (4) can be very close to 1 which makes GD algorithm converges very slowly. On the other hand, removing the top few PCs will make  $C$  in (5) significantly smaller than 1. In other words, GD may take a lot of iterations to converge when solving (1) directly while the second stage of LING takes much less iterations to converge. This can also be seen in the experiments of section 4.

## 3 Theorems

In this section we compute the risk of LING estimator under the fixed design setting. For simplicity, assume  $\mathbf{U}_1, \mathbf{D}_0$  generated by Algorithm 2 give exactly the top  $k_2$  left singular vectors and singular values of  $\mathbf{X}$  and GD in step 4 of Algorithm 1 converges to the optimal solution. Let  $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$  be the SVD of  $\mathbf{X}$  where  $\mathbf{U} = (\mathbf{U}_1, \mathbf{U}_2)$  and  $\mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2)$ . Here  $\mathbf{U}_1, \mathbf{V}_1$  are top  $k_2$  singular vectors and  $\mathbf{U}_2, \mathbf{V}_2$  are bottom  $p - k_2$  singular vectors. Let  $\mathbf{D} = \text{diag}(\mathbf{D}_1, \mathbf{D}_2)$  where  $\mathbf{D}_1 \in k_2 \times k_2$  contains top  $k_2$  singular values denoted by  $d_1 \geq d_2 \geq \dots \geq d_{k_2}$  and  $\mathbf{D}_2 \in p - k_2 \times p - k_2$  contains bottom  $p - k_2$  singular

values. Let  $\mathbf{D}_3 = \text{diag}(\mathbf{0}, \mathbf{D}_2)$  (replace  $\mathbf{D}_1$  in  $\mathbf{D}$  by a zero matrix of the same size). □

### 3.1 The Fixed Design Model

Assume  $\mathbf{X}$ ,  $\mathbf{Y}$  comes from the fixed design model  $\mathbf{Y} = \mathbf{X}\beta + \epsilon$  where  $\epsilon \in n \times 1$  are i.i.d Gaussian noise with variance  $\sigma^2$ . Note that  $\mathbf{X} = \mathbf{U}_1 \mathbf{D}_1 \mathbf{V}_1^\top + \mathbf{X}_r$ , the fixed design model can also be written as

$$\mathbf{Y} = (\mathbf{U}_1 \mathbf{D}_1 \mathbf{V}_1^\top + \mathbf{X}_r) \beta + \epsilon = \mathbf{U}_1 \gamma_1 + \mathbf{X}_r \gamma_2 + \epsilon$$

where  $\gamma_1 = \mathbf{D}_1 \mathbf{V}_1^\top \beta$  and  $\gamma_2 = \beta$ . We use the  $l_2$  distance between  $\mathbb{E}(\mathbf{Y}|\mathbf{X})$  (the best possible prediction given  $\mathbf{X}$ ) and  $\hat{\mathbf{Y}} = \mathbf{U}_1 \hat{\gamma}_{1,s} + \mathbf{X}_r \hat{\gamma}_2$  (the prediction by LING) as the loss function. The risk of LING can be written as

$$\begin{aligned} & \frac{1}{n} \mathbb{E} \|\mathbb{E}(\mathbf{Y}|\mathbf{X}) - \mathbf{U}_1 \hat{\gamma}_{1,s} - \mathbf{X}_r \hat{\gamma}_2\|^2 \\ &= \frac{1}{n} \mathbb{E} \|\mathbf{U}_1 \gamma_1 + \mathbf{X}_r \gamma_2 - \mathbf{U}_1 \hat{\gamma}_{1,s} - \mathbf{X}_r \hat{\gamma}_2\|^2 \end{aligned}$$

We can further decompose the risk into two terms:

$$\begin{aligned} & \frac{1}{n} \mathbb{E} \|\mathbf{U}_1 \gamma_1 + \mathbf{X}_r \gamma_2 - \mathbf{U}_1 \hat{\gamma}_{1,s} - \mathbf{X}_r \hat{\gamma}_2\|^2 = \\ & \frac{1}{n} \mathbb{E} \|\mathbf{U}_1 \gamma_1 - \mathbf{U}_1 \hat{\gamma}_{1,s}\|^2 + \frac{1}{n} \mathbb{E} \|\mathbf{X}_r \gamma_2 - \mathbf{X}_r \hat{\gamma}_2\|^2 \end{aligned} \quad (6)$$

because  $\mathbf{U}_1^\top \mathbf{X}_r = \mathbf{0}$ . Note that here the expectation is taken with respect to  $\epsilon$ .

Let's calculate the two terms in (6) separately. For the first term we have:

**Lemma 1.**

$$\frac{1}{n} \mathbb{E} \|\mathbf{U}_1 \gamma_1 - \mathbf{U}_1 \hat{\gamma}_{1,s}\|^2 = \frac{1}{n} \sum_{j=1}^{k_2} \frac{d_j^4 \sigma^2 + \gamma_{1,j}^2 n^2 \lambda^2}{(d_j^2 + n\lambda)^2} \quad (7)$$

Here  $\gamma_{1,j}$  is the  $j^{\text{th}}$  element of  $\gamma_1$ .

*Proof.* Let  $S \in k_2 \times k_2$  be the diagonal matrix with  $S_{j,j} = \frac{d_j^2}{d_j^2 + n\lambda}$ . So we have  $\hat{\gamma}_{1,s} = S \mathbf{U}_1^\top \mathbf{Y} = S \gamma_1 + S \mathbf{U}_1^\top \epsilon$ ,  $\mathbb{E}(\hat{\gamma}_{1,s}) = S \gamma_1$ .

$$\begin{aligned} & \frac{1}{n} \mathbb{E} \|\mathbf{U}_1 \gamma_1 - \mathbf{U}_1 \hat{\gamma}_{1,s}\|^2 \\ &= \frac{1}{n} \mathbb{E} \|\mathbf{U}_1 \mathbb{E}(\hat{\gamma}_{1,s}) - \mathbf{U}_1 \hat{\gamma}_{1,s}\|^2 \\ & \quad + \frac{1}{n} \|\mathbf{U}_1 \gamma_1 - \mathbf{U}_1 \mathbb{E}(\hat{\gamma}_{1,s})\|^2 \\ &= \frac{1}{n} \mathbb{E} \|\mathbf{U}_1 S \mathbf{U}_1^\top \epsilon\|^2 + \frac{1}{n} \|\gamma_1 - S \gamma_1\|^2 \\ &= \frac{1}{n} \mathbb{E} \text{Tr}(\mathbf{U}_1 S^2 \mathbf{U}_1^\top \epsilon \epsilon^\top) + \frac{1}{n} \|\gamma_1 - S \gamma_1\|^2 \\ &= \frac{1}{n} \mathbb{E} \text{Tr}(S^2) \sigma^2 + \frac{1}{n} \|\gamma_1 - S \gamma_1\|^2 \\ &= \frac{1}{n} \sum_{j=1}^{k_2} \frac{d_j^4 \sigma^2 + \gamma_{1,j}^2 n^2 \lambda^2}{(d_j^2 + n\lambda)^2} \end{aligned}$$

Now consider the second term in (6).

Note that

$$\mathbf{X}_r = \mathbf{U} \mathbf{D}_3 \mathbf{V}^\top$$

The residual  $\mathbf{Y}_r$  after the first stage can be represented by

$$\mathbf{Y}_r = \mathbf{Y} - \mathbf{U}_1 \hat{\gamma}_1 = (\mathbf{I} - \mathbf{U}_1 \mathbf{U}_1^\top) \mathbf{Y} = \mathbf{X}_r \gamma_2 + (\mathbf{I} - \mathbf{U}_1 \mathbf{U}_1^\top) \epsilon$$

and the optimal coefficient obtained in the second GD stage is

$$\hat{\gamma}_2 = (\mathbf{X}_r^\top \mathbf{X}_r + n\lambda \mathbf{I})^{-1} \mathbf{X}_r^\top \mathbf{Y}_r$$

For simplicity, let  $\epsilon_2 = (\mathbf{I} - \mathbf{U}_1 \mathbf{U}_1^\top) \epsilon$ .

**Lemma 2.**

$$\mathbb{E} \|\mathbf{X}_r \gamma_2 - \mathbf{X}_r \hat{\gamma}_2\|^2 = \sum_{i=k_2+1}^p \frac{1}{(d_i^2 + n\lambda)^2} (d_i^4 \sigma^2 + n\lambda^2 d_i^2 \alpha_i^2) \quad (8)$$

where  $\alpha_i$  is the  $i^{\text{th}}$  element of  $\alpha = \mathbf{V}^\top \gamma_2$

*Proof.* First define

$$\begin{aligned} \mathbf{X}_\lambda &= \mathbf{X}_r^\top \mathbf{X}_r + n\lambda \mathbf{I} \\ \mathbf{D}_\lambda &= \mathbf{D}_3^2 + n\lambda \mathbf{I} \end{aligned}$$

$$\begin{aligned} \mathbb{E} \|\mathbf{X}_r \gamma_2 - \mathbf{X}_r \hat{\gamma}_2\|^2 &= \|\mathbf{X}_r \gamma_2 - \mathbf{X}_r \mathbb{E}(\hat{\gamma}_2)\|^2 \\ &+ \mathbb{E} \|\mathbf{X}_r \mathbb{E}(\hat{\gamma}_2) - \mathbf{X}_r \hat{\gamma}_2\|^2 \end{aligned} \quad (9)$$

Consider (9) and (10) separately.

$$\begin{aligned} (9) &= \|\mathbf{X}_r \mathbf{X}_\lambda^{-1} \mathbf{X}_r^\top \mathbf{X}_r \gamma_2 - \mathbf{X}_r \gamma_2\|^2 \\ &= \|\mathbf{U} \mathbf{D}_3 \mathbf{D}_\lambda^{-1} \mathbf{D}_3^2 \mathbf{V}^\top \gamma_2 - \mathbf{U} \mathbf{D}_3 \mathbf{V}^\top \gamma_2\|^2 \\ &= \|\mathbf{D}_3 \mathbf{D}_\lambda^{-1} \mathbf{D}_3^2 \alpha - \mathbf{D}_3 \alpha\|^2 \\ &= \sum_{i=k_2+1}^p \alpha_i^2 d_i^2 \left( \frac{n\lambda}{d_i^2 + n\lambda} \right)^2 \end{aligned}$$

$$\begin{aligned} (10) &= \mathbb{E}_{\epsilon_2} \|\mathbf{X}_r \mathbf{X}_\lambda^{-1} \mathbf{X}_r^\top \epsilon_2\|^2 \\ &= \mathbb{E}_{\epsilon_2} \text{Tr}(\mathbf{X}_r \mathbf{X}_\lambda^{-1} \mathbf{X}_r^\top \mathbf{X}_r \mathbf{X}_\lambda^{-1} \mathbf{X}_r^\top \epsilon_2 \epsilon_2^\top) \\ &= \mathbb{E}_{\epsilon_2} \text{Tr}(\mathbf{D}_3 \mathbf{D}_\lambda^{-1} \mathbf{D}_3^2 \mathbf{D}_\lambda^{-1} \mathbf{D}_3 \mathbf{U}^\top \epsilon_2 \epsilon_2^\top \mathbf{U}) \\ &= \text{Tr}(\mathbf{D}_3 \mathbf{D}_\lambda^{-1} \mathbf{D}_3^2 \mathbf{D}_\lambda^{-1} \mathbf{D}_3 \mathbb{E}_{\epsilon_2} [\mathbf{U}^\top \epsilon_2 \epsilon_2^\top \mathbf{U}]) \end{aligned}$$

Note that

$$\mathbb{E}_{\epsilon_2} [\mathbf{U}^\top \epsilon_2 \epsilon_2^\top \mathbf{U}] = \text{diag}(0, I_{p-k_2}) \sigma^2$$

( $\text{diag}(0, I_{p-k_2})$  replace the top  $k_2 \times k_2$  block of the identity matrix with 0),

$$(10) = \sum_{i=k_2+1}^p \frac{d_i^4}{(d_i^2 + n\lambda)^2} \sigma^2 \quad (11)$$

Add the two terms together finishes the proof. □

Plug (7) (8) into (6) we have

**Theorem 2.** *The risk of LING algorithm under fixed design setting is*

$$\frac{1}{n} \sum_{j=1}^{k_2} \frac{d_j^4 \sigma^2 + \gamma_{1,j}^2 n^2 \lambda^2}{(d_j^2 + n\lambda)^2} + \frac{1}{n} \sum_{i=k_2+1}^p \frac{d_i^4 \sigma^2 + n^2 \lambda^2 d_i^2 \alpha_i^2}{(d_i^2 + n\lambda)^2} \quad (12)$$

**Remark 3.** *This risk is the same as the risk of ridge regression provided by Lemma 1 in (Dhillon et al., 2013). Actually, LING gets exactly the same prediction as RR on the training dataset. This is very intuitive since on the training set LING is essentially decomposing the RR solution into the first stage shrinkage PCR predictor on top  $k_2$  PCs and the second stage GD predictor over the residual spaces as explained in section 2.*

## 4 Experiments

In this section we compare the accuracy and computational cost (evaluated in terms of FLOPS) of 3 different algorithms for solving Ridge Regression: Gradient Descent with Optimal step size (GD), Stochastic Variance Reduction Gradient (SVRG) (Johnson and Zhang, 2013) and LING. Here SVRG is an improved version of stochastic gradient descent which achieves exponential convergence with constant step size. We also consider Principle Component Regression (PCR) (Artemiou and Li, 2009; Jolliffe, 2005) which is another common way for running large scale regression. Experiments are performed on 3 simulated models and 2 real datasets. In general, LING performs well on all 3 simulated datasets while GD, SVRG and PCR fails in some cases. For two real datasets, all algorithms give reasonable performance while SVRG and LING are the best. Moreover, both stages of LING only requires only a moderate amount of matrix multiplications each cost  $O(np)$ , much faster to run on matlab compared with SVRG which contains a lot of loops.

### 4.1 Simulated Data

Three different datasets are constructed based on the fixed design model  $\mathbf{Y} = \mathbf{X}\beta + \epsilon$  where  $\mathbf{X}$  is of size  $2000 \times 1500$ . In the three experiments  $\mathbf{X}$  and  $\beta$  are generated randomly in different ways (more details in following sections) and i.i.d Gaussian noise is added to  $\mathbf{X}\beta$  to get  $\mathbf{Y}$ . Then GD, SVRG, PCR and LING are performed on the dataset. For GD, we try different number of iterations  $n_1$ . For SVRG, we vary the number of passes through data denoted by  $n_{\text{SVRG}}$ . The number of iterations SVRG takes equals the number of passes through data times sample size and each iteration takes  $O(p)$  FLOPS. The step size of SVRG is chosen by cross validation but this cost is not considered when evaluating the total computational cost. Note that one advantage of GD and LING is that due to the simple quadratic form

Table 1: parameter setup for simulated data

	MODEL 1	MODEL 2	MODEL 3
$k_1$	21,22,23,26 30,50,100	20,30,50 100,150,400	20,30,50,100 150,400
$n_1$	10,20,30 50,80,100 150,200	2,4,6,8,10 15,20,30	6,10,15,20 30,50,80 120,180,250
$k_2$	20	20	20
$n_2$	1,2,3,5 8,13,20	2,4,6,8,10 15,20,30	2,4,6,8,10 15,30
$n_{\text{SVRG}}$	30,50,80 120,150	5,10,20 30,50	5,10,15,25 40,60,90

of the target function, their step size can be computed directly from the data without cross validation which introduces extra cost. For PCR we pick different number of PCs ( $k_1$ ). For LING we pick top  $k_2$  PCs in the first stage and try different number of iterations  $n_2$  in the second stage. The computational cost and the risk of the four algorithms are computed. The above procedure is repeated over 20 random generation of  $\mathbf{X}$ ,  $\beta$  and  $\mathbf{Y}$ . The risk and computational cost of the traditional RR solution (2) for every dataset is also computed as a benchmark.

The parameter set up for the three datasets are listed in table 1.

#### 4.1.1 Model 1

In this model the design matrix  $\mathbf{X}$  has a steep spectrum. The top 30 singular values of  $\mathbf{X}$  decay exponentially as  $1.3^i$  where  $i = 40, 39, 38, \dots, 11$ . The spectrum of  $\mathbf{X}$  is shown in figure 4. To generate  $\mathbf{X}$ , we fix the diagonal matrix  $\mathbf{D}_e$  with the designed spectrum and construct  $\mathbf{X}$  by  $\mathbf{X} = \mathbf{U}_e \mathbf{D}_e \mathbf{V}_e^\top$  where  $\mathbf{U}_e, \mathbf{V}_e$  are two random orthonormal matrices. The elements of  $\beta$  are sampled uniformly from interval  $[-2.5, 2.5]$ . Under this set up, most of the energy of the  $\mathbf{X}$  matrix lies in top PCs since the top singular values are much larger than the remaining ones so PCR works well. But as indicated by (4), the convergence of GD is very slow.

The computational cost and average risk of the four algorithms and also the RR solution (2) over 20 repeats are shown in figure 1. As shown in figure 1 both PCR and LING work well by achieving risk close to RR at less computational cost. SVRG is worse than PCR and LING but much better than GD.

#### 4.1.2 Model 2

In this model the design matrix  $\mathbf{X}$  has a flat spectrum. The singular values of  $\mathbf{X}$  are sampled uniformly from  $[\frac{\sqrt{2000}}{2}, \sqrt{2000}]$ . The spectrum of  $\mathbf{X}$  is shown in figure 5. To generate  $\mathbf{X}$ , we fix the diagonal matrix  $\mathbf{D}_e$  with the designed spectrum and construct  $\mathbf{X}$  by  $\mathbf{X} = \mathbf{U}_e \mathbf{D}_e \mathbf{V}_e^\top$  where  $\mathbf{U}_e, \mathbf{V}_e$  are two random orthonormal matrices. The ele-

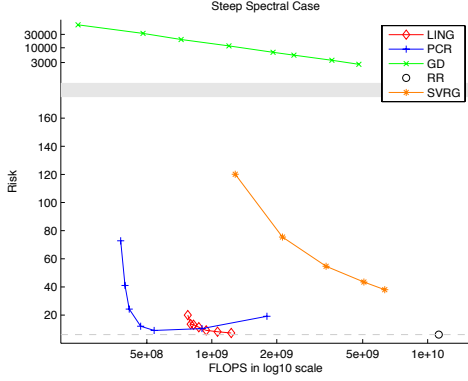


Figure 1: **Model 1:** Risk VS. Computational Cost plot. PCR and LING approaches the RR risk very fast. SVRG also approaches RR risk but cost more than the previous two. GD is very slow and inaccurate.

ments of  $\beta$  are sampled uniformly from interval  $[-2.5, 2.5]$ . Under this set up, the signal are widely spread among all PCs since the spectrum of  $\mathbf{X}$  is relatively flat. PCR breaks down because it fails to catch the signal on bottom PCs. As indicated by (4), GD converges relatively fast due to the flat spectrum of  $\mathbf{X}$ .

The computational cost and average risk of the four algorithms and also the RR solution (2) over 20 repeats are shown in figure 2. As shown by the figure GD works best since it approaches the risk of RR at the the lowest computational cost. LING and SVRG also works by achieving reasonably low risk with less computational cost. PCR works poorly as explained before.

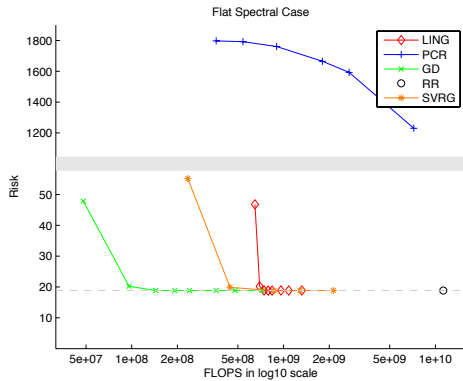


Figure 2: **Model 2:** Risk VS. Computational Cost plot. GD approaches the RR risk very fast. SVRG and LING are slower than GD but still achieves risk close to RR at less cost. PCR is slow and has huge risk.

### 4.1.3 Model 3

This model presented a special case where both PCR and GD will break down. The singular values of  $\mathbf{X} \in 2000 \times 1500$  are constructed by first uniformly sample from  $[\frac{\sqrt{2000}}{2}, \sqrt{2000}]$ . The top 15 sampled values are then multiplied by 10. The top 100 singular values of  $\mathbf{X}$  are shown in figure 6. To generate  $\mathbf{X}$ , we fix the diagonal matrix  $\mathbf{D}_e$  with the designed spectrum and construct  $\mathbf{X}$  by  $\mathbf{X} = \mathbf{U}_e \mathbf{D}_e$  where  $\mathbf{U}_e$  is a random orthonormal matrix. The first 15 and last 1000 elements of the coefficient vector  $\beta \in 1500 \times 1$  are sampled uniformly from interval  $[-2.5, 2.5]$  and other elements of  $\beta$  remains 0. In this set up,  $\mathbf{X}$  has orthogonal columns which are the PCs, and the signal lies only on the top 15 and bottom 1000 PCs. PCR won't work since a large proportion of signal lies on the bottom PCs. On the other hand, GD won't work as well since the top few spectral values are too large compared with other singular values, which makes GD converges very slowly.

The computational cost and risk of the four algorithms and also the RR solution (2) over 20 repeats are shown in figure 3. As shown by the figure LING works best in this set up. SVRG is slightly worse than LING but still approaching RR with less cost. In this case, GD converges slowly and PCR is completely off target as explained before.

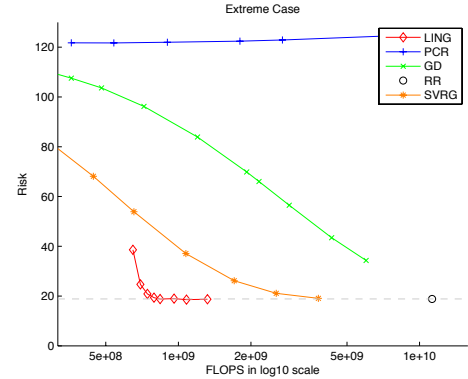


Figure 3: **Model 3:** Risk VS. Computational Cost plot. LING approaches RR risk the fastest. SVRG is slightly slower than LING. GD also approaches RR risk but cost more than LING. PCR has a huge risk no matter how many PCs are selected.

## 4.2 Real Data

In this section we compare PCR, GD, SVRG and LING with the RR solution (2) on two real datasets.

### 4.2.1 Gisette Dataset

The first is the gisette data set (Guyon, 2003) from the UCI repository which is a bi-class classification task. Every row

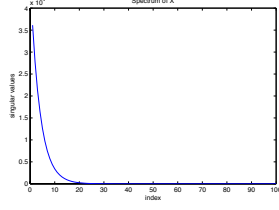


Figure 4: Top 100 singular values of  $\mathbf{X}$  in Model 1

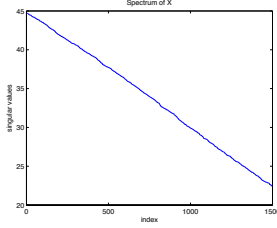


Figure 5: Singular values of  $\mathbf{X}$  in Model 2

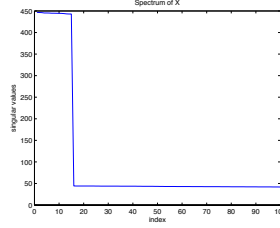


Figure 6: Top 100 singular values of  $\mathbf{X}$  in Model 3

of the design matrix  $\mathbf{X} \in 6000 \times 5000$  consists of pixel features of a single digit "4" or "9" and  $\mathbf{Y}$  gives the class label. Among the 6000 samples, we use 5000 for training and 1000 for testing. The classification error rate for RR solution (2) is 0.019. Since the goal is to compare different algorithms for regression, we don't care about achieving the state of the art accuracy for this dataset as long as regression works reasonably well. When running PCR, we pick top  $k_1 = 10, 20, 40, 80, 150, 300, 400$  PCs and in GD we iterate  $n_1 = 2, 5, 10, 15, 20, 30, 50, 100, 150$  times. For SVRG we try  $n_{\text{SVRG}} = 1, 2, 3, 5, 10, 20, 40, 80$  passes through the data. For LING we pick  $k_2 = 5, 15$  PCs in the first stage and try  $n_2 = 1, 2, 4, 8, 10, 15, 20, 30, 50$  iterations in the second stage. The computational cost and average classification error of the four algorithms and also the RR solution (2) on test set over 6 different train test splits are shown in figure 7. The top 150 singular values of  $\mathbf{X}$  are shown in figure 9. As shown in the figure, SVRG gets close to the RR error very fast. The two curves of LING with  $k_2 = 5, 15$  are slower than SVRG since some initial FLOPS are spent on computing top PCs but after that they approach RR error very fast. GD also converges to RR but cost more than the previous two algorithms. PCR performs worst in terms of error and computational cost.

#### 4.2.2 Buzz in Social Media

The second dataset is the UCI buzz in social media dataset which is a regression task. The goal is to predict popularity (evaluated by the mean number of active discussion) of a certain topic on Twitter over a period. The original feature matrix contains some statistics about this topic over that period like number of discussions created and new authors interacting at the topic. The original feature dimen-

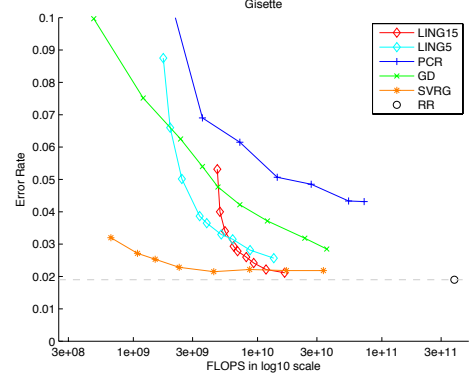


Figure 7: **Gisette:** Error Rate VS. Computational Cost plot. SVRG achieves small error rate fastest. Two LING lines with different  $n_2$  spent some FLOPS on computing top PCs first, but then converges very fast to a lower error rate. GD and PCR also provide reasonably small error rate and are faster than RR, but suboptimal compared with SVRG and LING.

sion is 77. We add quadratic interactions to make it 3080. To save time, we only used a subset of 8000 samples. The samples are split into 6000 train and 2000 test. We use MSE on the test data set as the error measure. For PCR we pick  $k_1 = 10, 20, 30, 50, 100, 150$  PCs and in GD we iterate  $n_1 = 1, 2, 4, 6, 8, 10, 15, 20, 30, 40, 60, 100$  times. For SVRG we try  $n_{\text{SVRG}} = 1, 2, 3, 5, 10, 15, 20, 40, 80$  passes through the dataset and for LING we pick  $k_2 = 5, 15$  in the first stage and  $n_2 = 0, 1, 2, 4, 6, 8, 10, 15, 20, 25$  iterations in the second stage. The computational cost and average MSE on test set over 5 different train test splits are shown in figure 8. The top 150 singular values of  $\mathbf{X}$  are shown in figure 10. As shown in the figure, SVRG approaches MSE of RR very fast. LING spent some initial FLOPS for computing top PCs but after that converges fast. GD and PCR also achieves reasonable performance but suboptimal compared with SVRG and LING. The MSE of PCR first decays when we add more PCs into regression but finally goes up due to overfit.

## 5 Summary

In this paper we present a two stage algorithm LING for computing large scale Ridge Regression which is both fast and robust in contrast to the well known approaches GD and PCR. We show that under the fixed design setting LING actually has the same risk as Ridge Regression assuming convergence. In the experiments, LING achieves good performances on all datasets when compare with three other large scale regression algorithms.

We conjecture that same strategy can be also used to accelerate the convergence of stochastic gradient descent when solving Ridge Regression since the first stage in LING es-

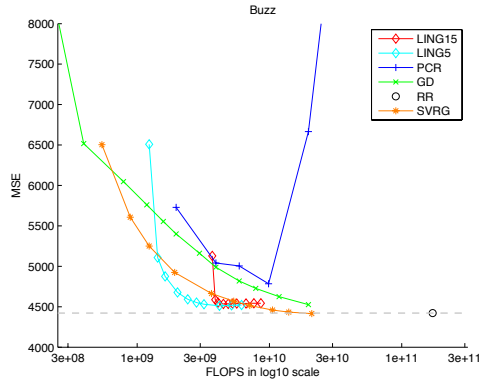


Figure 8: **Buzz** : MSE VS. Computational Cost plot. SVRG and two LING lines with different  $n_2$  achieves small MSE fast. GD is slower than LING and SVRG. PCR reaches its smallest MSE at  $k_1 = 50$  then overfits.

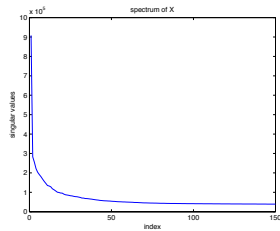


Figure 9: Top 150 singular values of  $\mathbf{X}$  in Gisette Dataset

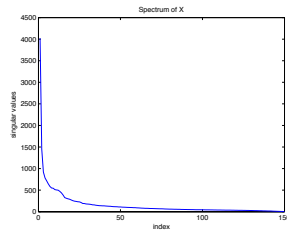


Figure 10: Top 150 singular values of  $\mathbf{X}$  in Social Media Buzz Dataset

entially removes the high variance directions of  $\mathbf{X}$ , which will lead to variance reduction for the random gradient direction generated by SGD.

## References

- Marina A. Epelman. Rate of convergence of steepest descent algorithm. 2007.
- Andreas Artemiou and Bing Li. On principal components and regression: a statistical explanation of a natural phenomenon. *Statistica Sinica*, 19(4):1557–1565, 2009. ISSN 1017-0405.
- Léon Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMP-STAT'2010)*, pages 177–187, Paris, France, August 2010. Springer.
- Paramveer S. Dhillon, Dean P. Foster, Sham M. Kakade, and Lyle H. Ungar. A risk comparison of ordinary least squares vs ridge regression. *Journal of Machine Learning Research*, 14:1505–1511, 2013.

Isabelle Guyon. Design of experiments for the nips 2003 variable selection benchmark. 2003.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, May 2011a. ISSN 0036-1445.

Nathan Halko, Per-Gunnar Martinsson, Yoel Shkolnisky, and Mark Tygert. An algorithm for the principal component analysis of large data sets. *SIAM J. Scientific Computing*, 33(5):2580–2594, 2011b.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Ian Jolliffe. *Principal Component Analysis. Encyclopedia of Statistics in Behavioral Science*. John Wiley & Sons, 2005.

Yichao Lu, Paramveer S. Dhillon, Dean Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

G. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proc. 15th International Conf. on Machine Learning*, pages 515–521. Morgan Kaufmann, San Francisco, CA, 1998.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *ICML 2004: Proceedings of the twenty-first International Conference on Machine Learning*. OMNIPRESS, pages 919–926, 2004.